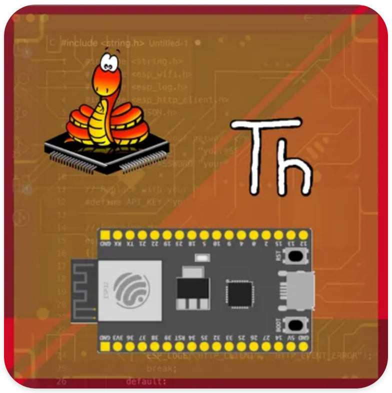
← Back to Blog



MicroPython on ESP32 and ESP8266 Getting Started

Learn how to set up MicroPython on your ESP32 with this guide. Flash firmware with Thonny IDE, write python scripts to interact with the ESP32 microcontroller.

November 24, 2024 Getting Started

Today, we're going to jump straight into getting MicroPython up and running on the ESP32. If you've ever wanted to simplify coding on microcontrollers, MicroPython is a game-changer. It lets you write and execute Python scripts directly on your ESP32 or ESP8266, making it quick and easy to prototype and bring your ideas to life.

In this post, we will Setup Thonny IDE, which makes working with MicroPython very user friedly, Flash MicroPython Firmware – prepare the ESP32 by installing the MicroPython runtime, and make our first MicroPython program running in ESP32.

Requirements

Before diving into MicroPython, here's what you'll need to get started. The great news is that MicroPython supports not only the original ESP32 chip but also a range of other variants, including:

- ESP8266

This means you can use almost any board based on these chips, whether it's a dev kit, custom board, or even modules. The flexibility of MicroPython ensures compatibility across the entire ESP32 and ESP8266 family.

Download and Install Thonny IDE

Thonny IDE is an excellent, beginner-friendly tool for writing and running Python code, and it comes with built-in support for MicroPython. It's lightweight, intuitive, and works seamlessly with ESP32 boards.

You can download Thonny IDE directly from the Thonny Official Website

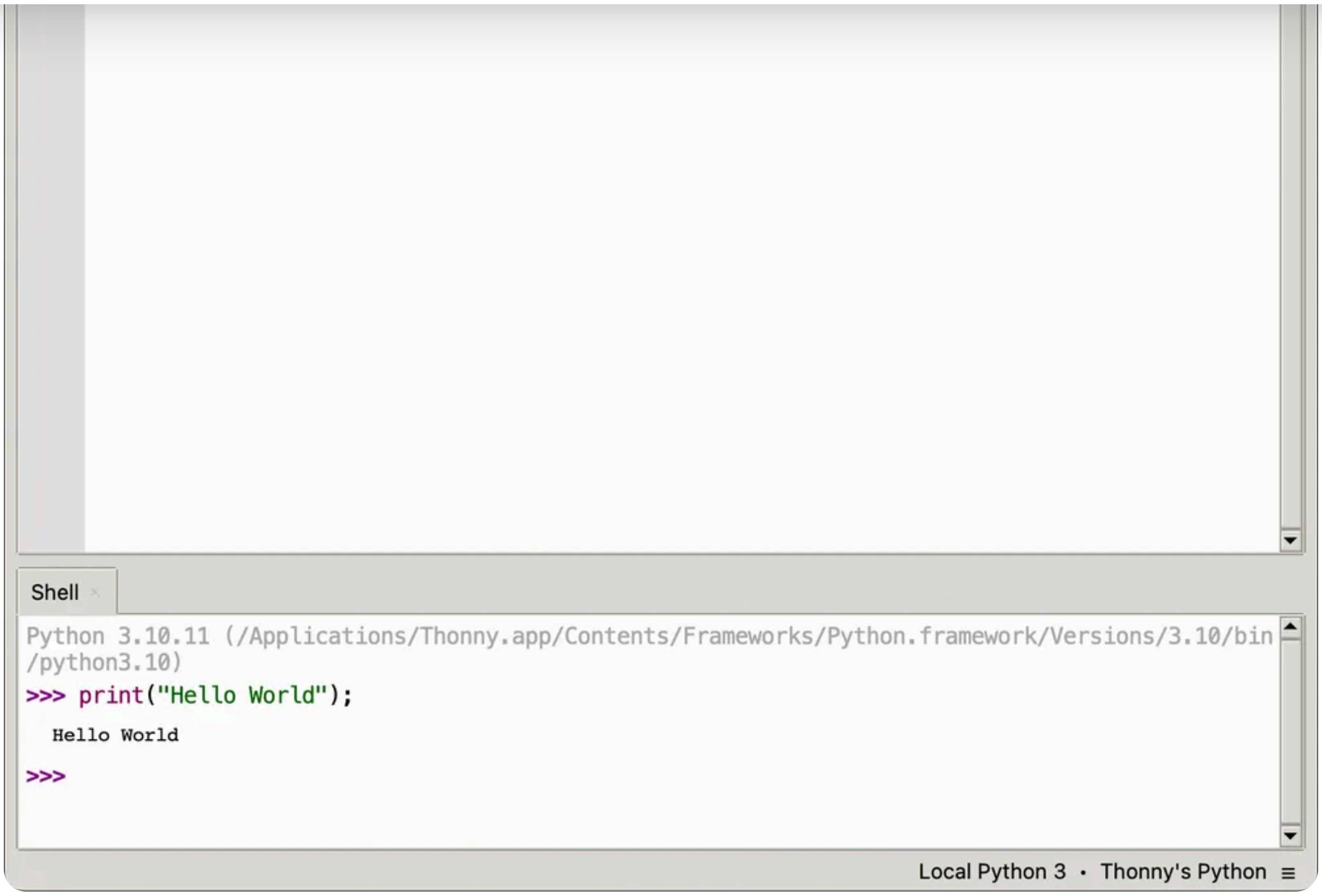
Installation Instructions: - **Windows**: Download the installer from the website and follow the on-screen instructions. - **macOS**: If you're using Homebrew, you can install it with the following command:

```
brew install --cask thonny
```

- **Linux**: Use your package manager or download it from the website. For example, on Ubuntu:

```
sudo apt update
sudo apt install thonny
```

Once you've installed Thonny IDE, launch the application. When opened for the first time, it should display a simple, beginner-friendly interface, typically looking like this:

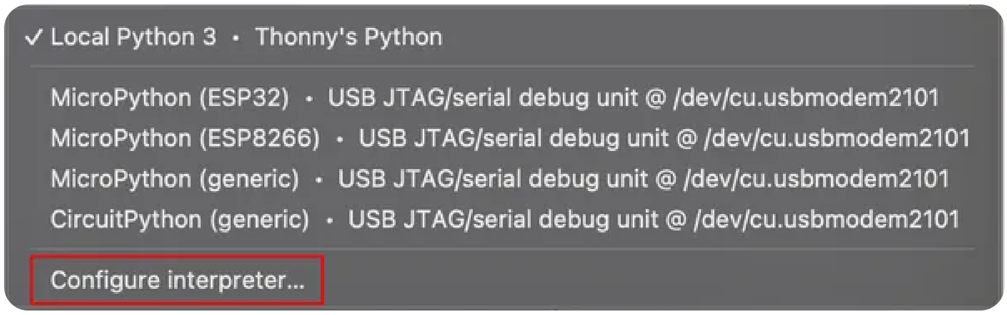


```
Python 3.10.11 (/Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.10/bin
/python3.10)
>>> print("Hello World");
  Hello World
>>>
```

Install MicroPython on ESP32

First things first - connect your ESP32 to your computer using a USB cable. Make sure the board is powered on and ready to go.

Look at the bottom-right corner of the Thonny IDE, click on the hamburger menu (three horizontal lines), from the dropdown menu, select "Configure Interpreter".

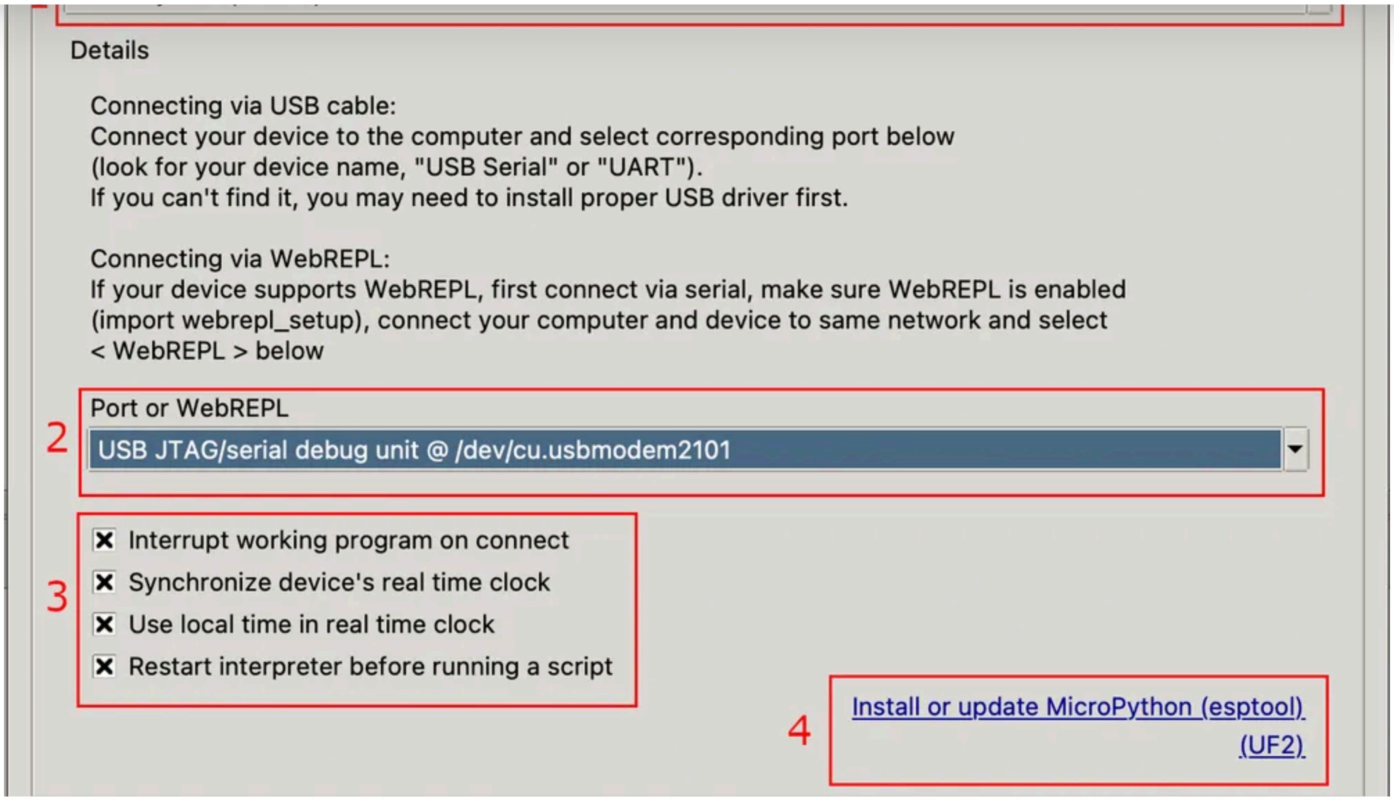

✓ Local Python 3 • Thonny's Python
MicroPython (ESP32) • USB JTAG/serial debug unit @ /dev/cu.usbmodem2101
MicroPython (ESP8266) • USB JTAG/serial debug unit @ /dev/cu.usbmodem2101
MicroPython (generic) • USB JTAG/serial debug unit @ /dev/cu.usbmodem2101
CircuitPython (generic) • USB JTAG/serial debug unit @ /dev/cu.usbmodem2101
Configure interpreter...

computer!

2. **Set the Port**: Under the Port dropdown, select the correct `COM port` (Windows) or `/dev/tty.*` device (macOS/Linux) that corresponds to your ESP32.
3. **Enable All Checkboxes**:

- Interrupt working program on connect
- Synchronize device's real time clock
- Use local time in real time clock
- Restart interpreter before running a script

4. Click on "**Install or update MicroPython (esptool)**" to flash the latest MicroPython firmware onto your ESP32.


Details

Connecting via USB cable:
Connect your device to the computer and select corresponding port below
(look for your device name, "USB Serial" or "UART").
If you can't find it, you may need to install proper USB driver first.

Connecting via WebREPL:
If your device supports WebREPL, first connect via serial, make sure WebREPL is enabled
(import webrepl_setup), connect your computer and device to same network and select
< WebREPL > below

2 Port or WebREPL: USB JTAG/serial debug unit @ /dev/cu.usbmodem2101

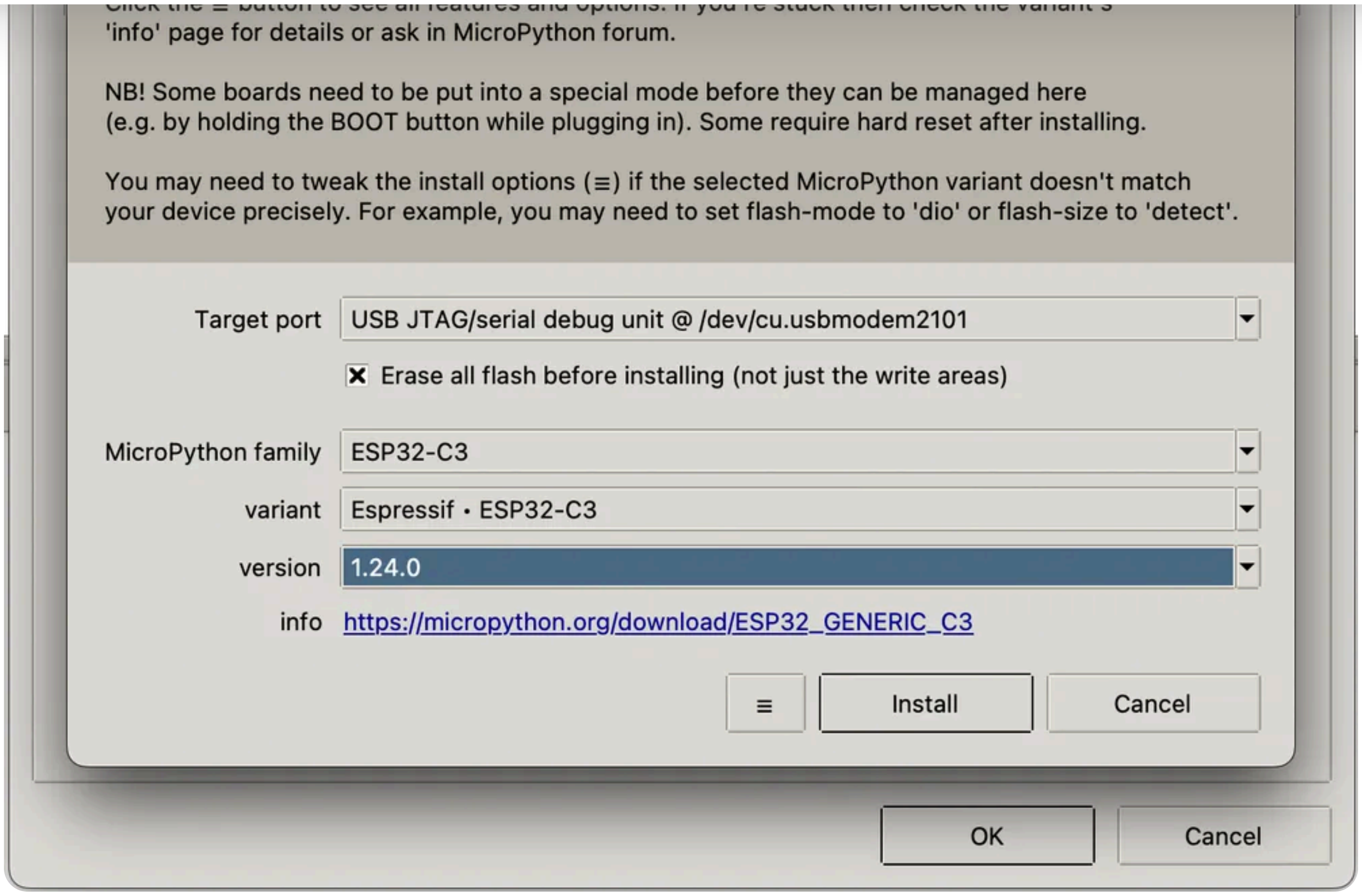
3
- [x] Interrupt working program on connect
- [x] Synchronize device's real time clock
- [x] Use local time in real time clock
- [x] Restart interpreter before running a script

4 Install or update MicroPython (esptool)
(UF2)

When the **Install or Update MicroPython** popup appears, follow these steps:

1. **MicroPython Family**: Select `ESP32` as the MicroPython family, we are using ESP32-C3 board, therefore we select `ESP32-C3`.
2. **Variant**: Choose the variant that corresponds to your specific board (e.g., `Espressif ESP32-C3`).
3. **Version**: Select the `latest` version, which is currently `1.24.0`.

Click `Install` and wait for the process to complete.

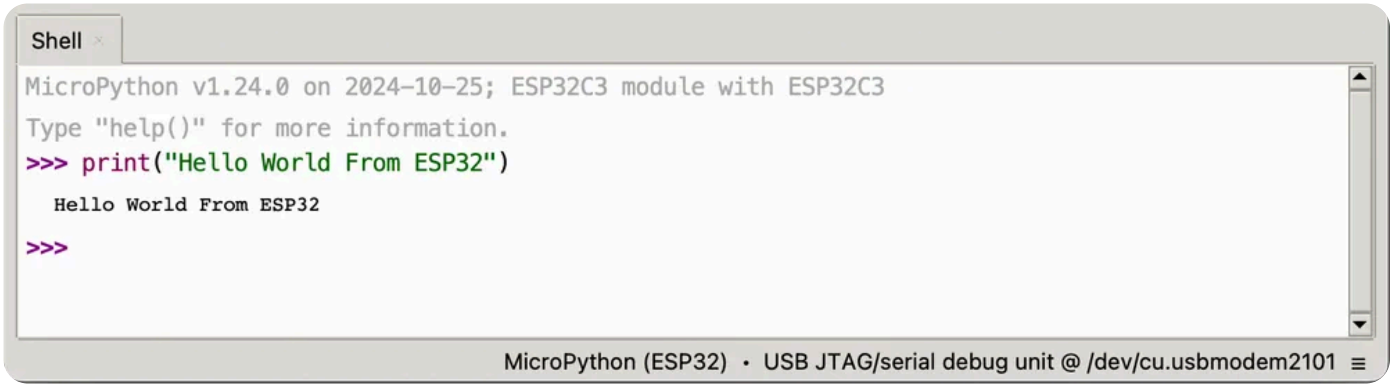

'info' page for details or ask in MicroPython forum.

NB! Some boards need to be put into a special mode before they can be managed here (e.g. by holding the BOOT button while plugging in). Some require hard reset after installing.

You may need to tweak the install options (≡) if the selected MicroPython variant doesn't match your device precisely. For example, you may need to set flash-mode to 'dio' or flash-size to 'detect'.

Target port	USB JTAG/serial debug unit @ /dev/cu.usbmodem2101
	[✓] Erase all flash before installing (not just the write areas)
MicroPython family	ESP32-C3
variant	Espressif • ESP32-C3
version	1.24.0
info	https://micropython.org/download/ESP32_GENERIC_C3

If everything is set up correctly, Thonny will connect to your ESP32 board, and you'll see a message in the console at the bottom of the IDE that looks like this:



```
MicroPython v1.24.0 on 2024-10-25; ESP32C3 module with ESP32C3
Type "help()" for more information.
>>> print("Hello World From ESP32")
  Hello World From ESP32
>>>
```

MicroPython (ESP32) • USB JTAG/serial debug unit @ /dev/cu.usbmodem2101

This indicates that your ESP32 board is successfully running MicroPython, and you can now start interacting with it directly from the console or by running scripts.

Testing the MicroPython on ESP32

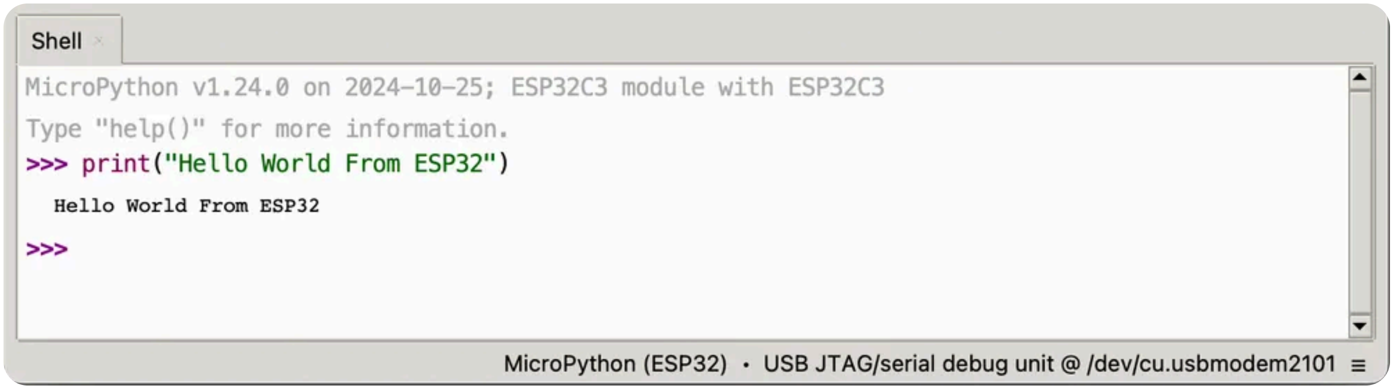
In the **script editor area** (upper part of Thonny IDE), type the following code:

```
from machine import Pin

p0 = Pin(10, Pin.OUT)
p0.on()
```

Note: The code uses GPIO10 as the built-in LED pin for this example. However, the built-in LED pin may vary depending on the specific ESP32 board you are using. Check your board information in ESP32 Boards

Once you've written the code in the script editor area in Thonny IDE, Click the **Play** button (green triangle) on the toolbar. Alternatively, go to the menu and select Run > Run Current Script.


Shell
MicroPython (ESP32) • USB JTAG/serial debug unit @ /dev/cu.usbmodem2101

```
MicroPython v1.24.0 on 2024-10-25; ESP32C3 module with ESP32C3
Type "help()" for more information.
>>> print("Hello World From ESP32")
  Hello World From ESP32
>>>
```

If everything is set up correctly, the built-in LED (on `GPIO10` in this example) will light up as the script runs.

Since we defined the built-in LED pin as `p0` in the script, we can now control it directly from the **Thonny shell**. To turn the LED ON again, type `p0.on()`.

MicroPython (ESP32) • USB JTAG/serial debug unit @ /dev/cu.usbmodem2101

One of the great advantages of working with MicroPython is the ability to directly interact with your ESP32 in real-time, without needing to re-upload the code every time. This is possible because once variables (like p0 in this example) and functions are defined in your script, they remain accessible in the shell as long as the script is running or loaded.

This is what makes MicroPython truly awesome - it enables quick prototyping, allowing you to test hardware functionality and tweak parameters instantly. Debugging becomes straightforward as you can experiment with different commands directly in the shell to see how your code interacts with the hardware. Additionally, it supports interactive development, letting you make changes and observe results in real time, without the need to repeatedly re-upload the firmware.

Other ESP32 Getting Started Guides

In case you are interested in other ESP32 Getting Started Guides, check:

- **ESP32 Getting Started with ESP-IDF in Visual Studio Code**
- **ESP32 Getting Started with Arduino IDE**

Conclusion

Getting started with MicroPython on the ESP32 opens up a world of possibilities for rapid prototyping and interactive development. With just a few simple steps, you can set up your board, write Python scripts, and directly control hardware without the need for complex setups or repetitive firmware uploads.

In this tutorial, we tested the ESP32 by running MicroPython code to control the built-in LED. We demonstrated how to write scripts, save them to the board, and interact with the hardware dynamically using the Thonny shell.